Practical Data Sovereignty Enforcement for Industrial Data Sharing with AssetHub

Zheng Hung Gan1, b), Thein Lai Wong1, a), Wah Pheng Lee2, c)

1 Tunku Abdul Rahman University of Management and Technology, Jalan Genting Kelang, Kuala Lumpur, 53300, Malaysia

2Omis Consulting Sdn. Bhd., Jalan Sungai Besi, Kuala Lumpur, 57100, Malaysia

a) Corresponding author: wongtl@tarc.edu.my  
b) ganzh-wr20@student.tarc.edu.my

*c) leewp@omis-consulting.com*

**Abstract.** Ensuring data sovereignty in Industry 4.0 environments requires more than restricting access to asset data. It also demands oversight of how the data is used after it has been shared. This paper introduces a lightweight and practical usage control mechanism that is embedded within AssetHub, a data-sharing framework based on the Asset Administration Shell. The mechanism applies a JSON-based policy model and includes runtime components such as the Policy Enforcement Point, the Policy Decision Point, and an Evaluator. It supports two enforcement methods that offer fine-grained control over instance data. One method performs transformations at the field level, while the other excludes records based on defined constraints. The solution is modular, imposes minimal overhead, and operates without reliance on semantic tooling, which makes it suitable for integration into responsive and extensible industrial systems.

# INTRODUCTION

As Industry 4.0 continues to reshape manufacturing and industrial collaboration, the exchange of asset data across organizational boundaries has become a key enabler of automation, predictive maintenance, and real-time decision-making. However, such data exchanges present a persistent challenge. Once data is accessed, traditional access control mechanisms offer no way to govern how it is used afterward. This limitation raises concerns around compliance, trust, and the safeguarding of intellectual property, especially in decentralized industrial ecosystems.

To address this, the concept of data sovereignty has become increasingly important. It focuses on ensuring that data usage aligns with the intent of the original provider. Several frameworks and data usage control models have been proposed in response, including policy-driven approaches such as those introduced by the International Data Spaces (IDS) initiative [1]. Despite their strengths, many of these solutions depend on complex semantic models or middleware-level enforcement, which makes them difficult to adopt in the lightweight, asset-centric platforms often used in practice.

This paper presents a practical solution that extends AssetHub, a modular data-sharing platform based on the Asset Administration Shell model, to support the runtime enforcement of Data Usage Control policies. The contribution introduces a lightweight policy description language expressed in JSON and a set of tailored policy patterns. These include techniques such as selective anonymization and conditional exclusion of records, which are designed to operate directly on asset instance data.

The effectiveness of this approach is demonstrated through two local experiments that validate enforcement correctness in real-time data-sharing scenarios. Unlike prior work, the solution offers fine-grained, per-asset policy enforcement without the semantic or computational overhead of existing frameworks. The following section outlines relevant background and related work in DUC, asset representation, and policy enforcement in industrial systems.

# Background and Related Work

## Industry 4.0 and Data Sharing

Industry 4.0 refers to the ongoing digital transformation of manufacturing and industrial ecosystems through the convergence of cyber-physical system, Internet of Things (IoT), and advanced data analytics [2,3]. These technologies support real-time process integration, system interconnectivity, and data-driven decision-making across organizational boundaries. At the core of this shift is the need to share asset-related data reliably between autonomous industrial actors. However, conventional access control mechanisms govern only initial access and do not provide guarantees on how data is subsequently used. The lack of control over data after access heightens the risks related to compliance, confidentiality, and the protection of intellectual property. In response, the idea of data sovereignty has become a central design concern, emphasizing the need to ensure that shared data is used according to the terms intended by the provider [1].

## Asset Administration Shell and AssetHub

The AAS is a standardised framework for digitally representing industrial assets, allowing properties, operations, and identity information to be described in a machine-readable format. It plays a foundational role in enabling semantic interoperability and data integration across Industry 4.0 platforms. Building on this concept, AssetHub was developed as a modular platform for decentralised asset data sharing. It retains the structured modelling approach of AAS but was originally focused on secure data access and asset registration [4]. However, AssetHub lacked mechanisms to govern how shared data could be used after access was granted, limiting its ability to enforce compliance with provider-defined constraints. This highlighted the need for an extended enforcement layer capable of runtime DUC aligned with asset-centric operations.

## DUC in Industrial Ecosystems

A wide range of DUC frameworks have been developed to complement access control by regulating how data may be used after access. The UCONABC model introduced the notion of ongoing control by supporting obligations, attribute mutability, and continuous authorization [5]. It provided a theoretical basis for dynamic policy enforcement but remained abstract in implementation.

Practical systems have since adapted these ideas to industrial and distributed environments. LUCON, proposed by Schütte and Brost [6], enforces data flow policies using dynamic taint tracking and trust labels in message-oriented middleware. It supports runtime policy evaluation across distributed services and integrates with frameworks like Apache Camel. However, LUCON’s enforcement is limited to coarse-grained message flows and lacks the ability to control usage within structured asset data or digital twin representations.

In parallel, CamFlow [7] extends usage control to the cloud infrastructure level by embedding enforcement mechanisms directly into the operating system through provenance tracking. Though effective for ensuring compliance in cloud-based systems, its system-level architecture is not suited to lightweight industrial platforms such as AssetHub.

Industry-specific adaptations, such as OPC-UCON [8], embed usage control directly into the OPC-UA protocol stack for real-time industrial automation. However, these implementations are tightly coupled to communication protocols and lack portability for integration into more general-purpose, asset-centric frameworks.

Across these efforts, three limitations frequently emerge: the reliance on semantic-heavy policy languages (e.g., RDF, XACML), the absence of fine-grained enforcement at the asset or field level, and architectural dependencies that reduce adaptability. These gaps underscore the need for a lightweight, runtime-capable enforcement model that can operate directly on instance-level asset data in decentralized data sharing systems like AssetHub.

## Limitations of Existing DUC Frameworks for AssetHub

The existing DUC frameworks, while valuable, are not well suited for direct integration into AssetHub’s asset-oriented design. Semantic-heavy policies increase processing overhead and are poorly aligned with real-time industrial responsiveness. Centralized policy engines and protocol-specific enforcement further constrain adaptability. More critically, most frameworks are designed to operate at the system or middleware level and lack the capacity to enforce policies over structured instance-level data, such as anonymizing fields or filtering records within asset data representations.

AssetHub requires an approach that enables decentralized, per-asset control without sacrificing responsiveness or compatibility with existing AAS-inspired architectures. This paper addresses that need by proposing a lightweight, runtime-enforced policy mechanism designed specifically for enforcement at the asset instance layer.

# Tailored DUC Enforcement in AssetHub

## Architectural Extension of AssetHub

To support runtime data usage enforcement, AssetHub was extended with three core components: the Policy Enforcement Point (PEP), the Policy Decision Point (PDP), and the DUC Evaluator. These modules operate sequentially to enable instance-level policy enforcement within the data-serving pipeline.

The Policy Enforcement Point intercepts outgoing API requests and passes them to the Policy Decision Point, which evaluates the applicable policy using contextual attributes including the requesting party, the asset identifier, and the intended purpose of access. The Data Usage Control Evaluator processes the policy logic and enforces actions in real time. These actions may involve modifying sensitive fields or filtering out records that do not meet policy conditions, before the final response is returned to the consumer. This enforcement process is modular and policy-driven, and it is integrated seamlessly into the architecture of AssetHub.

Figure 1 provides a high-level overview of the extended AssetHub architecture with DUC integration. It shows how modules such as the AAS Core, Data Exchange Framework, and external Identity Provider and Broker interact to support secure, policy-compliant data sharing. Within this framework, usage control is achieved through a dedicated DUC Module that ensures all shared data adheres to provider-defined policies.

## A diagram of a software system AI-generated content may be incorrect.

**FIGURE 1.** High-level architecture of AssetHub with integrated DUC modules. The architecture includes trust establishment via Identity Provider and Broker, policy enforcement via the DUC Module, and structured data exchange through the Connector and Real-time Hub

While Figure 1 outlines the overall architecture of AssetHub with integrated DUC capabilities, Figure 2 zooms into the internal workflow of the DUC module itself. It details the interaction among components responsible for policy authoring, registration, decision-making, and enforcement. The process begins at the Policy Administration Point (PAP), where data providers define usage policies. These are stored via the Policy Management Point (PMP) and retrieved during runtime by the Policy Decision Point (PDP) when triggered by a request intercepted at the Policy Enforcement Point (PEP). If additional context is needed, the Policy Information Point (PIP) supplies attributes such as user role or time of access. Once evaluated, the PDP returns a decision that the PEP enforces before responding to the consumer.

Together, these two views demonstrate how AssetHub enforces dynamic, instance-level control over shared data, ensuring policy compliance and data sovereignty without compromising architectural modularity.

## A diagram of a data exchange AI-generated content may be incorrect.

**FIGURE 2.** Detailed internal flow of DUC component interactions. Policies are authored (PAP), stored (PMP), evaluated (PDP), and enforced (PEP) using contextual input (PIP) before asset data is shared

## Policy Representation Using JSON-Based PDL

The enforcement mechanism is driven by a lightweight PDL written in JSON. The PDL defines usage constraints based on factors such as identity, purpose, or time, and links each constraint with a corresponding enforcement duty that is applied when the condition is satisfied.

The design prioritizes modularity and simplicity. Constraints and duties are expressed as structured key-value pairs, making the format both human-readable and machine-enforceable. The policy logic is aligned with AssetHub’s internal data model, allowing it to operate directly on instance-level data with minimal overhead.

This structure enables flexible enforcement logic that is suitable for decentralized industrial environments, while avoiding the processing costs associated with semantic-rich policy languages.

## Enforcement Patterns and Runtime Behavior

Two tailored enforcement patterns were developed to address common industrial data-sharing requirements:

* Selective Property Transformation Pattern: modifies sensitive asset properties-such as replacing identifiers or personal fields with neutral placeholders-based on contextual constraints. This allows privacy preservation without structural disruption.
* Constraint-Based Record Filtering Pattern: removes entire asset records from the response set if specific policy conditions are not met. This supports fine-grained filtering based on parameters such as consumer identity, access purpose, or geographic context.

These enforcement patterns are evaluated dynamically during each request and applied directly to the asset’s instance data. This approach enables post-access control without requiring any changes to the upstream data generation processes.

# Evaluation and Discussion

This section evaluates the enforcement mechanism integrated into AssetHub, focusing on two core aspects: verifying that policies are correctly enforced at runtime and validating the compatibility of the approach with AssetHub’s existing modular architecture. The evaluation is based on a sample policy scenario applied to real asset instance data and reflects typical usage conditions found in decentralized industrial data-sharing environments.

## Enforcement Behavior Validation

A representative dataset of asset records was used to demonstrate the policy enforcement logic. The scenario illustrates how field-level anonymization and conditional exclusion were dynamically evaluated and applied during runtime. While limited in scope, the output confirms that both tailored enforcement patterns were triggered correctly in response to the policy constraint, providing functional validation of the enforcement logic. The JSON policy structure is shown in Figure 3.

## A screenshot of a computer program AI-generated content may be incorrect.

**FIGURE 3.** JSON policy structure enforcing field-level anonymization of the Operator property and conditional exclusion of records with "Critical error" in the ErrorLogs field

When the policy was evaluated at runtime, the system correctly enforced both duties. Figure 4 presents a representative snapshot of a single asset record before and after anonymization.

To illustrate the full impact of the enforcement process across all six records, Table 1 summarizes how each record was handled based on its field values.

## A screenshot of a computer AI-generated content may be incorrect.

**FIGURE 4.** JSON data before and after enforcement showing anonymization of the Operator field

**TABLE 1.** Summary of policy enforcement effect across asset records

|  |  |  |  |
| --- | --- | --- | --- |
| **Machine ID** | **Operator** | **ErrorLogs** | **Enforcement Results** |
| MCH001 | John | Normal | Operator Anonymized |
| MCH002 | Alice | Critical Error | Record Excluded |
| MCH003 | Sarah | Warning | Operator Anonymized |
| MCH004 | David | Normal | Operator Anonymized |
| MCH005 | Claire | Normal | Operator Anonymized |
| MCH006 | Mike | Normal | Operator Anonymized |

To evaluate the runtime performance of enforcement, 100 repeated requests were issued through both the Provider and Consumer pipelines, with and without usage control enabled. As shown in Figure 5, average latency for Provider requests increased modestly from 23.11 ms to 23.79 ms, while Consumer-side requests increased from 1.63 ms to 3.02 ms.

## A graph of different colored bars AI-generated content may be incorrect.

**FIGURE 5.** Benchmark results comparing average response times for Provider and Consumer pipelines, with and without enforcement. Each result reflects 100 repeated request trials.

These results confirm that the enforcement mechanism introduces only minor overhead. The consistency in measured error and standard deviation across trials further supports the system’s responsiveness and suitability for interactive or near-real-time industrial applications.

## Discussion

The evaluation confirms that the tailored usage control patterns are enforced correctly and efficiently within the AssetHub pipeline. The system introduces minimal overhead while maintaining architectural modularity, making it suitable for responsive industrial data-sharing environments.

The JSON-based PDL was selected for its readability, simplicity, and alignment with the underlying asset structure. This design favors lightweight integration and human-interpretable rules over complex semantic models such as XACML or RDF. However, this trade-off does limit expressiveness: the current system does not yet support nested conditions, temporal constraints, or purpose-bound policies. Future work will focus on extending the PDL to accommodate these features without sacrificing runtime efficiency.

The present prototype was evaluated in a local deployment using single-threaded request simulations. Further testing under concurrent loads and distributed environments is needed to validate the system’s scalability. Additionally, runtime evaluator performance and potential bottlenecks under high-load scenarios remain open areas for investigation.

# CONCLUSION

This paper presented a practical DUC mechanism integrated into AssetHub; a modular data-sharing platform inspired by the Asset Administration Shell. The proposed approach enables data providers to enforce post-access constraints on asset data at runtime, supporting industrial data sovereignty without disrupting upstream systems or requiring heavyweight infrastructure.

The enforcement mechanism is driven by a lightweight, JSON-based policy language and supports two tailored patterns: field-level anonymization and conditional record exclusion. Evaluation results confirm that these policies are correctly interpreted and applied during live data requests, operating directly on asset instance data in response to contextual constraints.

By integrating enforcement at the request level, the mechanism preserves AssetHub’s decentralized and asset-centric design, ensuring compatibility with real-time industrial data-sharing workflows. Future work will focus on extending enforcement across distributed deployments, introducing richer policy types such as temporal or purpose-bound constraints, and evaluating integration with federated policy decision frameworks.

# References

1. A. Eitel et al., “Usage Control in the International Data Spaces,” *IDS Association White Paper*, 2021.
2. F. Zezulka, P. Marcon, I. Vesely, and O. Sajdl, “Industry 4.0 – An Introduction in the phenomenon,” IFAC-Pap. **49**(25), 8–12 (2016).
3. A.G. Frank, L.S. Dalenogare, and N.F. Ayala, “Industry 4.0 technologies: Implementation patterns in manufacturing companies,” Int. J. Prod. Econ. **210**, 15–26 (2019).
4. J.H. Hang, D.S. Charles, Z.H. Gan, S.K. Gan, Y.M. Lim, W.P. Lee, T.L. Wong, and C.P. Goh, “Constructing a Real-Time Value-Chain Integration Architecture for Mass Individualized Juice Production,” Information **13**(2), 56 (2022).
5. J. Park, and R. Sandhu, “The UCONABC usage control model,” ACM Trans. Inf. Syst. Secur. **7**(1), 128–174 (2004).
6. J. Schuette, and G.S. Brost, “LUCON: Data Flow Control for Message-Based IoT Systems,” in *2018 17th IEEE Int. Conf. Trust Secur. Priv. Comput. Commun. 12th IEEE Int. Conf. Big Data Sci. Eng. Trust.*, (IEEE, New York, NY, USA, 2018), pp. 289–299.
7. T.F.J.-M. Pasquier, J. Singh, D. Eyers, and J. Bacon, “CamFlow: Managed Data-sharing for Cloud Services,” IEEE Trans. Cloud Comput. **5**(3), 472–484 (2017).
8. F. Martinelli, O. Osliak, P. Mori, and A. Saracino, “Improving security in industry 4.0 by extending OPC-UA with usage control,” in *Proc. 15th Int. Conf. Availab. Reliab. Secur.*, (ACM, Virtual Event Ireland, 2020), pp. 1–10.